



## Speeding Up ROS training For Technical team

---

Over the several years, Robot Operating System (ROS) is growing faster than ever. Apart from labs, there are an increasing number of commercial sector, industrial and services robots using ROS. Gradually, it has become a widely-used platform in the robotics research community and is becoming the essential skills for robotics engineers.

But as you may understand, robotics engineers spend a lot of time developing ROS based software. Sometimes, they need to gather more knowledge about a specific ROS subject. Sometimes the team needs to incorporate more engineers with ROS knowledge.

Typical option for ROS learning is the ROS Wiki tutorials. Hence the technical staffs will pass the days and weeks trying to get the most of it.

Here **we propose a smooth learning path for your team in order to maximize their ROS learning speed and get the best possible results.** This path is inspired by the book The first 20 hours, and basically consists of four steps:

---

---

## ROBOT IGNITE ACADEMY

1. DECONSTRUCT ROS
2. REMOVE STUFF
3. LEARN ROS
4. PRACTICE (A LOT)

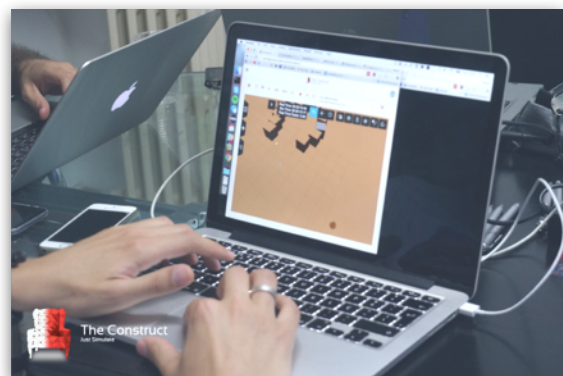
---

### 1. DECONSTRUCT ROS

---

Deconstructing ROS means to identify the different parts that compose ROS. This work has to be done by somebody that already knows ROS. We have done that work and identified the following main parts:

1. Installation and setup
2. Basic organization of the development environment with ROS (catkin workspaces, compilers issues, CMakeLists, IDE configuration)
3. Basic subjects: packages, roscore, rosparam server
4. Topics: publishers, subscribers, messages, how to create your own messages
5. Services: clients, servers, service messages, how to create your own service messages
6. Actions: clients, servers, service messages, how to create your own action messages
7. Debugging tools: rviz, rqt\_console, rqt\_graph, rosbags
8. TF
9. How to make a robot navigate: mapping, localization, path planning and obstacle avoidance
10. How to make a robot perceive: blob detection with OpenCV, object recognition, point cloud usage, people detection and recognition
11. Gazebo simulations
12. URDF robot creation
13. Robot Control
14. How to make a robot manipulate objects: MoveIt! usage, combining perception with manipulation, grasping



The previous points cover a global knowledge of ROS. This does not mean that your technical staffs will have to learn it all. Those points just express the most

---

## ROBOT IGNITE ACADEMY

---

common subjects required for the creation of an autonomous robot (like for example

the ones used in the Robocup@home competition).

---

## 2. REMOVE STUFF

---

That is, we eliminate as many things as possible, things that are not really necessary right now. The idea is to get the 80% of the results with the 20% of the effort.

### C++ OR PYTHON?

For the moment, we eliminate the necessity to use C++ for ROS. **Using C++ in ROS introduces three problems:**

1. First, the C++ version of ROS includes a lot more of concepts, like the `node_handle`, the `callback_queue` or having to deal with the threads of each callback (if you want them in parallel). Python handles all that by itself.
2. Second, by using Python, the learner will have to know just the minimum of CMake handling. Making the proper `CMakeLists.txt` in C++ is a nightmare. Instead, in Python you almost no need to touch the default one.
3. Using C++ for ANYTHING, makes the development a lot harder and by hence slower. And here speed is the key. The faster you can close the loop of doing something and experiencing the result, the faster the brain of the learner will make the connection that makes him learn the concept. With C++, compilation problems are of the first order.

I know you must be thinking: but C++ is our development language!! They need to learn in that language.

I understand

However, **starting their development in C++ is a bad idea, because it makes the pill to swallow a lot bigger.** Even if you need the technical staff to use ROS with C++, I recommend you to start with Python (even if the learner doesn't know about Python!!!)

Unless the technical staff is a master of C++, the following path:

Learning Python -> Learning ROS with Python -> Learning ROS with C++

is faster than the path:

Learning ROS with C++

Even if the learner already know C++, the additional amount of knowledge required to get ROS using that language makes the evolution very slow.

---

## ROBOT IGNITE ACADEMY

### INSTALLATION

The learner doesn't need to know about how to install. Starting with the installation is a waste of time, since at that point the learner knows nothing about ROS and may have trouble about the installation. ROS installation is a stupid step, that has nothing to do with ROS or intelligence or knowledge. It is just an experience, that the learner will be able to do faster when he already knows ROS.

So we suggest you avoid that step to your technical staffs.

### HOW TO CREATE A CATKIN\_WS

Again, this is a concept that is difficult to grasp if you don't know ROS and understand the problems of programming with it. Trying to make the learner understand why he has to create a catkin\_ws, how to do it and where, is a waste of time. The learner will not understand and keep coming to this point once and every time, slowing the progress speed.

### OTHER POINTS TO REMOVE

Other points to avoid are, configuring IDEs for programming with ROS (you must provide it done), what is ROS (who cares), how to setup Gazebo (provide it). Basically, you have to remove barriers, physical, emotional and mental, that make the work of learning ROS a lot harder.



**So in this point what we propose is that you have a system already set up and working for the technical staff, and that he concentrates on learning ROS for Python.**

---

## 3. LEARN ROS

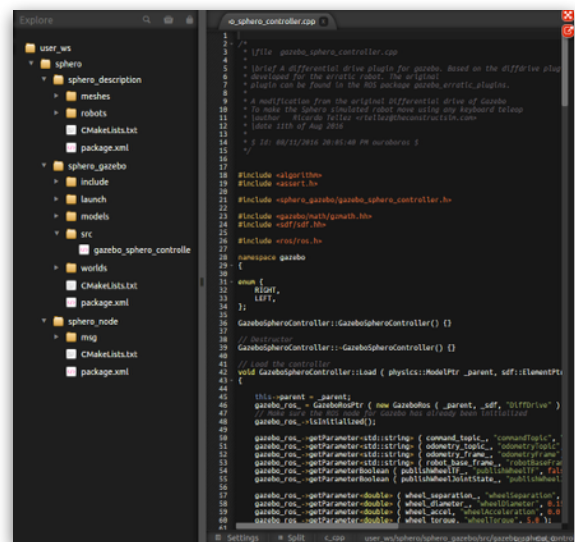
---

Here is where the learner has to dedicate the

time to learn. But **the learning has to be**

We propose you to provide to your technical staffs with the following sequence of tutorials of ROS (from [wiki.ros.org](http://wiki.ros.org)), which we have found to be optimal for faster learning:

11. Defining custom messages
12. Recording and playing back data
13. Visualization tutorials
14. TF
15. Gazebo
16. URDF
17. ROS control
18. ROS Navigation
19. PCL with ROS
20. MoveIt!



#### 4. PRACTICE (A LOT)

simulation, so the learner can see the results of his efforts quickly and with a meaning. Providing just a number example, for example a topic publishing a text, is not good enough to get the learner engaged. And engagement here is key if we want the

---

## ROBOT IGNITE ACADEMY

learner to learn quickly. The learner will be a lot more motivated if he can see the result of his efforts in a robot. We recommend to provide simulated robots, since they provide the possibility of testing quickly.

Practice should include an exam, if possible. Technical staffs learn the most under the stress situation of an evaluation. Sorry, but it works like that (I did not make the rules).

**Super important:** in order to apply the method and speed up learning, the learner

has to be practicing at least 20 hours in a row. Dedicating 20 hour focused in the important points will create a momentum that will increase the speed of learning.











## ROBOT IGNITE ACADEMY

### SUMMARY

So how can you have your technical staffs up to speed with ROS fast? This is a summary of what we have said above:

1. Provide a fully setup environment. Complete ROS + Gazebo + development IDE installed computer.
2. Provide the learner with the optimal list of tutorials to follow (either the one above of your own), but do not just point the learner to the ROS wiki.
3. Provide a full list of exercises that the learner has to solve, without providing the solution.
4. I know that all that is a lot of work but if you want to invest on that once, you will speed up the process of integration of new staffs in your team, and your results will raise.

Another option that you have is to use the services of Robot Ignite Academy. In our academy we provide everything already done for your team, organised in the proposed manner, including development environments, robot simulations, exercises and exams. Everything already working, and requiring only a web browser. No installation required, any computer will work. Give it a try!

|  |  |  |
|--|--|--|
|  <p><b>Robot Creation with URDF</b></p> <p>Advancing ROS</p> <p>Learn how to create the URDF files to control your robot with ROS</p>         |  <p><b>ROS Autonomous Vehicles 101</b></p> <p>Advancing ROS</p> <p>Introduction to Autonomous Vehicles in the ROS ecosystem</p>   |  <p><b>ROS-Industrial 101</b></p> <p>Advancing ROS</p> <p>Introduction of some basic ROS tools to control industrial robots with ROS</p> |
|  <p><b>OpenAI Gym for Robotics 101</b></p> <p>Advancing ROS</p> <p>Learn what is needed to be able to use OpenAI-Gym in your next project</p> |  <p><b>RTAB-Map in ROS 101</b></p> <p>Advancing ROS</p> <p>Learn how to use the rtabmap_ros package for performing RGB-D SLAM</p> |  <p><b>ROS Control 101</b></p> <p>Advancing ROS</p> <p>Learn how to ROSify the control of your robot</p>                                 |

